AIE425 Intelligent Recommender Systems, Fall Semester 24/25

Assignment #1: Neighborhood CF models (user, item-based CF)

221101107, Abdelrahman Mohamed Mahmoud Rizk

1. **Neighborhood-Based Collaborative Filtering**

1.1 overview

This analysis explores the development of a neighborhood-based recommender system using collaborative filtering (CF) techniques. By centering on user-based and item-based CF, the study details data collection, preprocessing, and structuring within a user-item matrix to enable similarity assessment. The objective of employing metrics such as Cosine and Pearson correlations is to identify patterns within user preferences to generate personalized and relevant recommendations.

1.2 Foundation  
  
In today’s digital world, recommender systems are essential, guiding users to discover content and products across streaming services, e-commerce platforms, and more and more. Collaborative filtering (CF) has become a popular method for generating recommendations by analyzing user interaction and preference patterns.

The process begins with collecting data through the web, which is then transformed into a rating matrix as explained in the lectures. Then we use Similarity measures, such as Cosine and Pearson correlations, to find clusters of users or items with overlapping preferences.

**2. Assignment requirements and description**  
 2.1 Companies use recommendation systems   
   
 Answer:  
   
 1. YouTube  
  
 2. MovieLens  
   
 3. Pinterest  
   
 4. TikTok  
  
 5. Alibaba

**2.2 Data Source for the Assignment**  
 Answer:  
  
 Movielens 25M Dataset [Movielens Dataset](https://grouplens.org/datasets/movielens/25m)  
  
**2.3 User feedback and rating type used**   
   
 Answer:

User feedback on movieLens is collected through movie reviews. The rating type used is 5-point interval rating.

This dataset describes 5-star rating from MovieLens, a movie recommendation service. It contains 32000204 ratings and 2000072 tag applications across 87585 movies. These data were created by 200948 users. Users were selected at random for inclusion. All selected users had rated at least 20 movies. Each user is represented by an ID, and no other information is provided

**2.4 Preprocessing, Cleaning, and Feedback**

Answer:

The dataset was loaded from a CSV file named "Rating.csv". This file contains user interaction data, including unique user and item identifiers, rating scores, and timestamps of each rating.

1. Dataset Overview:
   * The dataset initially contained 4 columns: UserId, movieId, Rating, and TimeStamp.
   * The TimeStamp column was dropped to reduce the dataset
   * The dataset now includes 3 columns: UserId, Movieid, and Rating.
   * Total Users: 200948
   * Total Movies: 87585
2. Total Ratings: 32000204  
     
   Rating Distribution:
   * Minimum Rating: 0.5
   * Maximum Rating: 5
   * rating distribution
     + 5 Stars: 152,646
     + 4.5 Stars: 90,051
     + 4 Stars: 280,291
     + 3.5 Stars: 130,755
     + 3 Stars: 207,514
     + 2.5 Stars: 53,093
     + 2 Stars: 69,617
     + 1.5 Stars: 17,494
     + 1 Star: 31,975
     + 0.5 Stars: 152,646
3. Missing Values Check:
   * UserId: 0 missing values
   * MovieId: 0 missing values
   * Rating: 0 missing values  
     Luckily there is no missing values!

2.5 User-Item Matrix

Answer: MovieId

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| UserId | M296 | M318 | M356 | M593 | M2571 |
| U3 | 0 | 4.0 | 4.0 | 0 | 4.0 |
| U13 | 5.0 | 4.5 | 0 | 4.5 | 4.5 |
| U19 | 3.5 | 0 | 4 | 4 | 4 |
| U23 | 5.0 | 5.0 | 0 | 5.0 | 1 |
| U26 | 3.0 | 4.5 | 3.0 | 0 | 5 |

2.6 Matrix Dataset Description

Answer:   
The dataset we have is a subset of the Movielens dataset which is a movie recommender system that uses 5 point interval rating (stars) so it was easier to gather the data from

Matrix Overview

1. Users (Rows):

* The matrix includes 5 users (U3, U13, U19, U23, and U26), these users has been selected because they did rate the same 5 movies

1. Items (Columns):
   * The matrix includes 5 Movies (M296, M318, M356, M593, M2571),
   * Each item represents a Movie where the number beside the \*M\* represents the movie ID from the original dataset.
2. Ratings:
   * The ratings in the matrix from 1 to 5, representing
3. Zero Values:
   * Each row (user) has some zero values, indicating unrated Movies by each user.

**2.7 CF Algorithms Background Overview and Analytical Solution**

Answer:

Collaborative Filtering (CF) is a method widely used in recommendation systems to predict the preferences of a user by collecting preferences from similar users or items. It's based on the idea that if two users have shown similar preferences, they may continue to do so in the future. CF is essential in applications like movie or product recommendations**,** as seen on platforms like Netflix,Tiktok,Pinterest, Amazon, and Spotify.

Types of Collaborative Filtering

1. User-Based Collaborative Filtering

- This approach finds users who have similar tastes or preferences to a target user. Recommendations are generated based on the ratings or interactions of these similar users.

- For instance, if User A and User B both enjoy a similar set of movies, a movie that User B liked but User A hasn’t seen yet might be recommended to User A (the example refers to the dataset we are typically using )

- Steps Involved:

1. Compute similarity scores between users based on past ratings or interactions.

2. Select a group of similar users to the target user.

3. Aggregate the ratings or interactions of these similar users to predict the target user’s interest.

2. Item-Based Collaborative Filtering

- Instead of focusing on users, this method finds items similar to those a target user has liked or interacted with.

- For example, if a user has rated a certain action movie highly, they may receive recommendations for other action movies based on item similarity.

- Steps Involved:

1. Compute similarity scores between items.

2. Find similar items for each item a user has rated.

3. Recommend items that are similar to what the user has previously shown interest in.

Key Algorithms in Collaborative Filtering

1. Memory-Based CF (Neighborhood-Based CF)

- Uses historical data directly to compute similarities between users or items.

- Common similarity measures include Cosine similarity, Pearson correlation, and Jaccard similarity.

Pros: Simple and interpretable; often effective for small datasets.

Cons: Struggles with scalability and sparsity in large datasets.

**User-Based Collaborative Filtering Analytical Approach:**

* Cosine similarity measure:

Predictions:

* Pearson correlation coefficient:

Predictions:

* + - Calculate the neighbors' bias and aggregate their ratings, weighted by similarity, to predict the target user's rating.

**2.8 Computing the average rating**

Average Ratings= =

2.9 **Computing the similarities using Cosine similarity and Pearson correlation coefficient**

Answer:

User 23 will be used as the target.

User-User CF:

Cosine Similarity CF User-based:

MovieId

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User | M296 | M318 | M356 | M593 | M2571 | Cosine (u23,M) |
| U3 | ? | 4.0 | 4.0 | ? | 4.0 | 0.832 |
| U13 | 5.0 | 4.5 | ? | 4.5 | 4.5 | 0.922 |
| U19 | 3.5 | ? | 4 | 4 | 4 | 0.873 |
| U23 | 5.0 | 5.0 | ? | 5.0 | 1 | 1 |
| U26 | 3.0 | 4.5 | 3.0 | ? | 5 | 0.807 |

For Movie 356:

* User 19's rating is 4.
* User 26's rating is 3.

Based on cosine similarity, we predict that **User 23's rating for Movie 356 will be approximately 3.5.**

Pearson Coefficient CF User-based:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| User-movie | M296 | M318 | M356 | M593 | M2571 | Mean Rating | Pearson (U1, i) |
| U3 | ? | 4.0 | 4.0 | ? | 4.0 | 4 | 0 |
| U13 | 5.0 | 4.5 | ? | 4.5 | 4.5 | 4.6 | 0.13 |
| U19 | 3.5 | ? | 4 | 4 | 4 | 3.8 | -0.1 |
| U23 | 5.0 | 5.0 | ? | 5.0 | 1 | 4 | 1 |
| U26 | 3.0 | 4.5 | 3.0 | ? | 5 | 3.8 | -0.7 |

Calculate Mean Ratings for Each User

To calculate the Pearson coefficient, we need to compute the mean (average) rating for each user, ignoring any missing values.

* **User 3 mean**:
* **User 13 mean**:
* **User 19 mean**:
* **User 23 mean:**
* **User 26 mean:**

### Fill in the Matrix with Mean-Adjusted Ratings

Now, we’ll adjust each rating by subtracting the user’s mean rating. The adjusted ratings matrix will look like this:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User-movie | M296 | M318 | M356 | M593 | M2571 |
| U3 | ? | 4-4=0 | 4-4=0 | ? | 4-4=0 |
| U13 | 5-4.6=0.4 | 4.5-4.6=-0.1 | ? | 4.5-4.6=-0.1 | 4.5-4.6=-0.1 |
| U19 | 3.5-3.8=-0.3 | ? | 4-3.8=0.2 | 4-3.8=0.2 | 4-3.8=0.2 |
| U23 | 5-4=1 | 5-4=1 | ? | 5-4=1 | 1-4=-3 |
| U26 | 3-3.8=-0.8 | 4.5-3.8=0.7 | 3-3.8=-0.8 | ? | 5-3.8=1.2 |

0.13

1. Overview and Intuition

* + - Cosine Similarity: Measures the angle between two vectors in a multidimensional space. In collaborative filtering, it treats user (or item) ratings as vectors and measures the similarity based on the direction of these vectors rather than their magnitude. It considers the overall pattern of ratings rather than their absolute values.
* Pearson Correlation Coefficient: Measures the linear relationship between two variables, adjusting for differences in the mean rating of each user. This metric considers how two users’ ratings deviate from their average rating and evaluates the strength and direction of this deviation. It’s useful for capturing how well two users follow a similar rating pattern relative to their personal baselines.

Key Difference: Cosine similarity doesn’t adjust for different average rating levels of users, while Pearson correlation does.

3. Pros and Cons of Cosine Similarity

Pros

* + - * Simple and Efficient: Cosine similarity is computationally less expensive and easier to understand. It only requires multiplying ratings and summing up values, without needing to subtract means.
* No Mean Adjustment Required: It’s effective when we’re only interested in the general direction (pattern) of ratings, rather than specific differences in rating scales.
  + - * + Works Well with Sparse Data: Cosine similarity can handle sparse data well since it only requires common items to compute similarity between users.

Cons

* + - * + Ignores Rating Scale Differences: It doesn’t account for users who may consistently rate higher or lower than others. For instance, if User A rates all movies 1 star higher than User B, cosine similarity may still see them as dissimilar.
        + Limited Sensitivity to Patterns: Since cosine similarity focuses on the direction, it may not capture nuanced relationships as effectively as Pearson correlation when users have different baseline ratings.

4. Pros and Cons of Pearson Correlation Coefficient

Pros

* + - * + Adjusts for Rating Scale Differences: By centering ratings around the mean, Pearson correlation accounts for differences in rating scales between users. This makes it more suitable when users have individual rating biases (e.g., a user who rates everything lower or higher than average).
        + Better at Capturing Linear Relationships: Pearson correlation captures linear trends, making it effective for finding users who rate items similarly relative to their own baselines.

Cons

* + - * + Sensitive to Outliers: Pearson correlation can be affected by outliers, especially if a user gives an unusually high or low rating to an item.
        + Requires at Least Two Ratings: To calculate Pearson correlation, we need at least two overlapping ratings between users. This can be a limitation in sparse datasets.
        + More Computationally Intensive: The formula for Pearson correlation involves more steps (like mean subtraction and squaring values), making it slightly more computationally expensive than cosine similarity.

5. When to Use Each Method

* + - * + Cosine Similarity is generally a good choice when:
        + You’re working with sparse datasets and want a straightforward, fast calculation.
        + The recommendation system only needs a rough similarity measure without fine-tuning based on individual rating biases.
        + The dataset has users with relatively consistent rating styles (e.g., no extreme rating variations).
        + Pearson Correlation is preferable when:
        + You want to adjust for different rating scales and baseline biases across users.
        + You have a denser dataset where users rate a similar set of items, allowing for more accurate correlation calculations.
        + You need to capture linear relationships where users may rate items relative to their personal baseline.

In summary:

* + - * + Cosine similarity is faster and simpler, focusing on general rating patterns without accounting for personal rating biases.
        + Pearson correlation is more accurate, adjusting for user-specific rating tendencies but requires more common items and is slightly more complex.

Item-Item CF:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User-movie | M296 | M318 | M356 | M593 | M2571 | Mean Rating |
| U3 | ? | 4.0 | 4.0 | ? | 4.0 | 4 |
| U13 | 5.0 | 4.5 | ? | 4.5 | 4.5 | 4.6 |
| U19 | 3.5 | ? | 4 | 4 | 4 | 3.8 |
| U23 | 5.0 | 5.0 | ? | 5.0 | 1 | 4 |
| U26 | 3.0 | 4.5 | 3.0 | ? | 5 | 3.8 |

Mean Centered Matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User-movie | M296 | M318 | M356 | M593 | M2571 |
| U3 | ? | 0 | 0 | ? | 0 |
| U13 | 0.4 | -0.1 | ? | -0.1 | -0.1 |
| U19 | -0.3 | ? | 0.2 | 0.2 | 0.2 |
| U23 | 1 | 1 | ? | 1 | -3 |
| U26 | -0.8 | 0.7 | -0.8 | ? | 1.2 |

0.22

**Assignment Results**

1. Cosine Similarity (User-User CF)

Measures similarity based on the cosine of the angle between users' rating vectors. The calculation involves summing the products of ratings and dividing by the square root of the sum of squares for each user's ratings.

|  |  |
| --- | --- |
| User Pair | Cosine Similarity |
| U23, U3 | 0.832 |
| U23, U13 | 0.922 |
| U23, U19 | 0.873 |
| U23, U26 | 0.807 |

2. Pearson Coefficient (User-User CF)

.

|  |  |
| --- | --- |
| User Pair | Pearson Coefficient |
| U23, U3 | 0 |
| U23, U13 | 0.13 |
| U23, U19 | -0.1 |
| U23, U26 | -0.7 |

3. Adjusted Cosine Similarity (Item-Item CF)

|  |  |
| --- | --- |
| Item Pair | Adjusted Cosine Similarity |
| M356, m318 | 1 |
| M356, m2571 | -1 |

4. Comparative Analysis

|  |  |  |
| --- | --- | --- |
| Method | User/Item Pair | Similarity Value |
| Cosine Similarity | U23, U3 | 0.832 |
| Pearson Coefficient | U23, U3 | 0 |
| Adjusted Cosine | M356, m318 | 1 |

**Brief about the Implementation Process, Tools, and Libraries Documentation**

1. Core Libraries Used

pandas as pd

os

numpy as np

from scipy.spatial.distance import cosine

from scipy.stats import pearsonr

- pandas (pd): Used for data manipulation and DataFrame operations

- os: Handles file path operations and directory management

- numpy (np): Supports numerical computations and array operations

- scipy: Provides scientific computing tools

- spatial.distance.cosine`: For cosine similarity calculations

-stats.pearsonr: For Pearson correlation computations

2. Implementation Process

2.1 Data Preprocessing

(file\_path, keep='first')

Key Features:

- Reads MovieLens ratings dataset from CSV

- Removes duplicate entries using `drop\_duplicates`

- Fills missing values with 0 using `fillna`

- Provides preprocessing statistics:

- Original row count

- Post-duplicate removal count

- Number of filled missing values

2.2 Matrix Creation

Process:

1. Identifies items rated by minimum number of users

2. Filters users who have rated minimum number of items

3. Creates user-item rating matrix using `pivot`

4. Returns both matrix and filtered DataFrame

2.3 Similarity Computations

Features:

- Supports both user-based and item-based similarity

- Implements two similarity measures:

1. Cosine Similarity

2. Pearson Correlation

- Returns similarity matrices as pandas DataFrames

2.4 Peer Group Formation

threshold=0.5

Process:

- Groups similar users/items based on threshold

- Removes self-references from peer groups

- Returns dictionary of peer groups

3.2 Data Storage

- Saves matrices to CSV files:

- User cosine similarity

- User Pearson correlation

- Item cosine similarity

- Item Pearson correlation

4. Implementation Method

4.1 Similarity Calculations

Two approaches implemented:

1. Cosine Similarity

- Measures angle between user/item vectors

- Range: [-1, 1]

- Implemented using scipy's cosine distance

2. Pearson Correlation

- Measures linear correlation

- Range: [-1, 1]

- Implemented using scipy's pearsonr

**2.12 Remarks about User-Based and Item-Based CF using Pearson Coefficient**

1. User-Based Collaborative Filtering (CF):

* Core Focus: Identifies users with similar taste patterns
* Primary Mechanism:

Finds users who rate items similarly

Particularly effective when users have overlapping interests

Works best when number of users < number of items

Using Pearson Correlation:

- Advantages:

Considers rating patterns relative to user averages

Compensates for different rating scales (high vs. low raters)

Captures personal preferences effectively

- Disadvantages:

Scales poorly with large user bases

Requires real-time calculations

Sensitive to sparse data

2. Item-Based Collaborative Filtering:

- Core Focus: Identifies items with similar rating patterns

- Primary Mechanism:

Finds relationships between items based on user ratings

Especially useful with sparse user ratings

More effective when users have rated few items

Using Pearson Correlation:

- Advantages:

More stable (item relationships change less frequently)

Allows pre-computation of similarities

Better handles sparse data

More scalable overall

- Disadvantages:

May miss some personalization aspects

Requires initial computation overhead

3. Key Comparative Insights:

Pearson Correlation Approach:

- User-Based:

Examines common items between users

Considers rating patterns relative to personal averages

Helps identify similar tastes despite different rating scales

- Item-Based:

Compares ratings of items by common users

Identifies items that tend to be liked/disliked together

Focuses on item relationships

Cosine Similarity Approach:

- User-Based:

Measures similarity through rating vector angles

Focuses on rating direction rather than exact scores

Effective for finding similar preferences despite scale differences

- Item-Based:

Compares items based on user rating patterns

Identifies related items through rating relationships

Useful for finding items often liked together

- Stability:

User preferences tend to change more frequently

item relationships are generally more stable

- Computation:

Item-based allows for pre-computation

User-based often needs real-time calculations  
  
**3. Conclusion**

**In this assignment, we examined various collaborative filtering (CF) methods to evaluate their effectiveness in item recommendation. The findings indicated that Cosine Similarity was effective for user-based CF in identifying similar users and generating recommendations based on high similarity scores. However, its performance was constrained by sparse data and variations in users' rating behaviors.**

**The Pearson Coefficient, which accounts for each user's average rating, provided better insights for users with diverse rating patterns. While it effectively addressed user biases, it faced difficulties with sparse data, particularly in cases where limited common ratings hindered its ability to accurately measure similarity.**

**Adjusted Cosine Similarity, used for item-based CF, tackled user biases by normalizing ratings, making it beneficial for item-item comparisons. This approach was especially effective in identifying items with strong similarities, although it encountered challenges with sparse datasets, occasionally leading to negative similarity values.**

**Reference**<https://grouplens.org/datasets/movielens/25m/> **this is the link of the dataset because the file is too big to upload on github**